CLAIMS

What is claimed is:

1. A method for avoiding reading z-values in a graphics pipeline, comprising:

(a) storing near z-values each representative of a near z-value on an object in a region, wherein the region is defined by a tile and a coverage mask therein;

(b) comparing the stored near z-values with far z-values computed for other objects in the region; and

(c) based on the comparison, conditionally reading from memory z-values previously stored for image samples in the region;

(d) wherein the comparison indicates whether an object is visible in the region.

2. The method as recited in claim 1, wherein the near z-values are stored in a record associated with the tile.

3. The method as recited in claim 1, wherein each near z-value represents a nearest z-value on the object in the region.

4. The method as recited in claim 1, wherein each far z-value represents a farthest z-value on the other objects in the region.

5. The method as recited in claim 1, wherein the previously stored z-values are read from memory only if the far z-values computed for the other objects in the region are farther than or equal to the corresponding near z-values.

6. The method as recited in claim 1, wherein a pair of the near z-values are stored for the tile.

7. The method as recited in claim 6, wherein a first near z-value is associated with a first sub-region covered by the coverage mask and a second near z-value is associated with a second sub-region not covered by the coverage mask.

8. The method as recited in claim 1, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against a first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and a renderer downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

9. A computer program product for avoiding reading z-values in a graphics pipeline, comprising:

(a) computer code for storing near z-values each representative of a near z-value on an object in a region, wherein the region is defined by a tile and a coverage mask therein;

(b) computer code for comparing the stored near z-values with far z-values computed for other objects in the region; and

(c) computer code for based on the comparison, conditionally reading from memory z-values previously stored for image samples in the region;

(d) wherein the comparison indicates whether an object is visible in the region.

10. The computer program product as recited in claim 9, wherein the near z-values are stored in a record associated with the tile.

11. The computer program product as recited in claim 9, wherein each near z-value represents a nearest z-value on the object in the region.

12. The computer program product as recited in claim 9, wherein each far z-value represents a farthest z-value on the other objects in the region.

13. The computer program product as recited in claim 9, wherein the previously stored z-values are read from memory only if the far z-values computed for the other objects in the region are farther than or equal to the corresponding near z-values.

14. The computer program product as recited in claim 9, wherein a pair of the near z-values are stored for the tile.

15. The computer program product as recited in claim 14, wherein a first near z-value is associated with a first sub-region covered by the coverage mask and a second near z-value is associated with a second sub-region not covered by the coverage mask.

16. The computer program product as recited in claim 9, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against a first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and a renderer downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which

were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

17. A system for avoiding reading z-values in a graphics pipeline, comprising:

(a) logic for storing near z-values each representative of a near z-value on an object in a region, wherein the region is defined by a tile and a coverage mask therein;

(b) logic for comparing the stored near z-values with far z-values computed for other objects in the region; and

(c) logic for, based on the comparison, conditionally reading from memory z-values previously stored for image samples in the region;

(d) wherein the comparison indicates whether an object is visible in the region.

18. A method for conservative stencil culling in a graphics pipeline, comprising:

(a) maintaining stencil values for regions at a plurality of levels of an image pyramid including a finest level and one or more coarser levels;

(b) determining whether the stencil value for a region at one of the coarser levels is valid; and

(c) performing conservative stencil culling on the region utilizing the stencil value at the coarser level if the stencil value at the coarser level is valid.

19. The method as recited in claim 18, wherein the stencil value for the region at the coarser level is representative of a plurality of portions of a corresponding region at a finer one of the levels of the image pyramid.

20. The method as recited in claim 19, wherein the stencil value for the region at the coarser level is valid if the stencil values of each of the portions of the corresponding region at the finer level are the same as each other.

21. The method as recited in claim 18, wherein the stencil value for the region at the coarser level is valid if the stencil values of each of a plurality of portions of a corresponding region at a finer level are the same as each other.

22. The method as recited in claim 21, wherein a valid flag is used to indicate whether the stencil value at the coarser level is valid.

23. The method as recited in claim 18, wherein the stencil value for the region at the coarser level is determined by reading stencil values from a corresponding region at the finest level.

24. The method as recited in claim 18, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against a first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and a rendering stage downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the rendering stage to be occluded.

25. The method as recited in claim 24, wherein the culling stage determines stencil value validity and the stencil values for the region at the coarser level using state information associated with the objects.

26. The method as recited in claim 25, wherein valid stencil values for the regions at the coarser levels are passed from the culling stage to the rendering stage.

27. A computer program product for conservative stencil culling in a graphics pipeline, comprising:

(a) computer code for maintaining stencil values for regions at a plurality of levels of an image pyramid including a finest level and one or more coarser levels;

(b) computer code for determining whether the stencil value for a region at one of the coarser levels is valid; and

(c) computer code for performing conservative stencil culling on the region utilizing the stencil value at the coarser level if the stencil value at the coarser level is valid.

28. The computer program product as recited in claim 27, wherein the stencil value for the region at the coarser level is representative of a plurality of portions of a corresponding region at a finer one of the levels of the image pyramid.

29. The computer program product as recited in claim 29, wherein the stencil value for the region at the coarser level is valid if the stencil values of each of the portions of the corresponding region at the finer level are the same as each other.

30. The computer program product as recited in claim 27, wherein the stencil value for the region at the coarser level is valid if the stencil values of each of a plurality of portions of a corresponding region at a finer level are the same as each other.

31. The computer program product as recited in claim 30, wherein a valid flag is used to indicate whether the stencil value at the coarser level is valid.

32. The computer program product as recited in claim 27, wherein the stencil value for the region at the coarser level is determined by reading stencil values from a corresponding region at the finest level.

33. The computer program product as recited in claim 27, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against a first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and a rendering stage downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the rendering stage to be occluded.

34. The computer program product as recited in claim 33, wherein the culling stage determines stencil value validity and the stencil values for the region at the coarser level using state information associated with the objects.

35. The computer program product as recited in claim 34, wherein valid stencil values for the regions at the coarser levels are passed from the culling stage to the rendering stage.

36. A system for conservative stencil culling in a graphics pipeline, comprising:

(a) logic for maintaining stencil values for regions at a plurality of levels of an image pyramid including a finest level and one or more coarser levels;

(b) logic for determining whether the stencil value for a region at one of the coarser levels is valid; and

(c) logic for performing conservative stencil culling on the region utilizing the stencil value at the coarser level if the stencil value at the coarser level is valid.

37. A method for creating a data structure adapted for use during conservative stencil culling, comprising:

(a) maintaining stencil values for regions at a plurality of levels of an image pyramid;

(b) determining whether all of the stencil values of a region at a finer one of the levels of the image pyramid are the same as each other; and

(c) storing a valid indicator indicating whether all of the stencil values of the region at the finer level are the same as each other;

(d) if all of the stencil values at the finer level are the same as each other, storing the stencil value.

38. The method as recited in claim 37, wherein the valid indicator and the stencil value are adapted for use during conservative stencil culling.

39. A computer program product for creating a data structure adapted for use during conservative stencil culling, comprising:

(a) computer code for maintaining stencil values for regions at a plurality of levels of an image pyramid;

(b) computer code for determining whether all of the stencil values of a region at a finer one of the levels of the image pyramid are the same as each other; and

(c) computer code for storing a valid indicator indicating whether all of the stencil values of the region at the finer level are the same as each other;

(d) computer code for, if all of the stencil values at the finer level are the same as each other, storing the stencil value.

40. The computer program product as recited in claim 39, wherein the valid indicator and the stencil value are adapted for use during conservative stencil culling.

41. A system for creating a data structure adapted for use during conservative stencil culling, comprising:

(a) logic for maintaining stencil values for regions at a plurality of levels of an image pyramid;

(b) logic for determining whether all of the stencil values of a region at a finer one of the levels of the image pyramid are the same as each other; and

(c) logic for storing a valid indicator indicating whether all of the stencil values of the region at the finer level are the same as each other;

(d) logic for, if all of the stencil values at the finer level are the same as each other, storing the stencil value.

42. A data structure stored in memory for use during conservative stencil culling, comprising:

(a) a valid indicator object indicating whether all stencil values of a region are equal to each other;

(b) a stencil value object including a stencil value equal to the stencil values of the region if all of the stencil values of the region are equal to each other.

43. A method for multiple-pass rendering using conservative occlusion culling, comprising:

(a) during a first pass,

passing objects from an input stream to a geometric processor for being transformed, and

sending the objects to a culling stage for creating an occlusion image in a first depth buffer requiring a first amount of storage; and

(b) during a second pass,

sending the objects to the culling stage for conservatively culling occluded objects utilizing the occlusion image, and

passing to a renderer remaining objects, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage.

44. The method as recited in claim 43, wherein shading operations are performed only during the second pass.

45. The method as recited in claim 43, wherein the first amount of storage is less than or equal to 1/2 the second amount of storage.

46. The method as recited in claim 43, wherein the first amount of storage is less than or equal to 1/4 the second amount of storage.

47. The method as recited in claim 43, wherein the first amount of storage is less than or equal to 1/8 the second amount of storage.

48. The method as recited in claim 43, wherein the culling stage has an input for receiving a plurality of the objects, the culling stage testing the objects against the first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and the renderer, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

49. A system for multiple-pass rendering using conservative occlusion culling, comprising:

(a) logic for a first pass including:

logic for passing objects from an input stream to a geometric processor for being transformed, and

logic for sending the objects to a culling stage for creating an occlusion image in a first depth buffer requiring a first amount of storage; and

(b) logic for a second pass including:

logic for sending the objects to the culling stage for conservatively culling occluded objects utilizing the occlusion image, and

logic for passing to a renderer remaining objects, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage.

50. The system as recited in claim 49, wherein shading operations are performed only during the second pass.

51. The system as recited in claim 49, wherein the first amount of storage is less than or equal to 1/2 the second amount of storage.

52. The system as recited in claim 49, wherein the first amount of storage is less than or equal to 1/4 the second amount of storage.

53. The system as recited in claim 49, wherein the first amount of storage is less than or equal to 1/8 the second amount of storage.

54. The system as recited in claim 49, wherein the culling stage has an input for receiving a plurality of the objects, the culling stage testing the objects against the first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and the renderer, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

55. A computer program product for multiple-pass rendering using conservative occlusion culling, comprising:

(a) computer code for a first pass including:

computer code for passing objects from an input stream to a geometric processor for being transformed, and

computer code for sending the objects to a culling stage for creating an occlusion image in a first depth buffer requiring a first amount of storage; and

(b) computer code for a second pass including:

computer code for sending the objects to the culling stage for conservatively culling occluded objects utilizing the occlusion image, and

computer code for passing to a renderer remaining objects, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage.

56. A method for multiple-pass rendering in a plurality of pipelines using conservative occlusion culling, comprising:

(a) during a first pass in a first pipeline,

passing objects from an input stream to a geometric processor of the first pipeline for being transformed, and

sending the objects to a culling stage of the first pipeline for creating an occlusion image in a first depth buffer requiring a first amount of storage; and

(b) during a second pass in a second pipeline,

sending the objects to a culling stage of the second pipeline for conservatively culling occluded objects utilizing the occlusion image, and

passing to a renderer remaining objects, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage.

57. The method as recited in claim 56, wherein the first and second pipelines operate in parallel.

58. The method as recited in claim 56, wherein the first and second pipelines operate on separate frames simultaneously.

59. The method as recited in claim 57, wherein the second pipeline includes a geometric processor for transforming objects.

60. The method as recited in claim 56, wherein the culling stage has an input for receiving a plurality of the objects, the culling stage testing the objects against the first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and the renderer, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

61. A system for multiple-pass rendering in a plurality of pipelines using conservative occlusion culling, comprising:

(a) logic for a first pass in a first pipeline including:

logic for passing objects from an input stream to a geometric processor of the first pipeline for being transformed, and

logic for sending the objects to a culling stage of the first pipeline for creating an occlusion image in a first depth buffer requiring a first amount of storage; and

(b) logic for a second pass in a second pipeline including:

logic for sending the objects to a culling stage of the second pipeline for conservatively culling occluded objects utilizing the occlusion image, and

logic for passing to a renderer remaining objects, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage.

62. The system as recited in claim 61, wherein the first and second pipelines operate in parallel.

63. The system as recited in claim 61, wherein the first and second pipelines operate on separate frames simultaneously.

64. The system as recited in claim 62, wherein the second pipeline includes a geometric processor for transforming objects.

65. The system as recited in claim 61, wherein the culling stage has an input for receiving a plurality of the objects, the culling stage testing the objects against the first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and the renderer, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

66. A computer program product for multiple-pass rendering in a plurality of pipelines using conservative occlusion culling, comprising:

(a) computer code for a first pass in a first pipeline including:

computer code for passing objects from an input stream to a geometric processor of the first pipeline for being transformed, and

computer code for sending the objects to a culling stage of the first pipeline for creating an occlusion image in a first depth buffer requiring a first amount of storage; and

(b) computer code for a second pass in a second pipeline including:

computer code for sending the objects to a culling stage of the second pipeline for conservatively culling occluded objects utilizing the occlusion image, and

computer code for passing to a renderer remaining objects, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage.

67. A method for avoiding reading z-values in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) during a first pass,

transforming objects,

creating an occlusion image in a first depth buffer requiring a first amount of storage, and

storing near z-values in the occlusion image, each near z-value representative of a near z-value on one of the objects; and

(b) during a second pass,

conservatively culling objects utilizing the occlusion image, and

rendering the remaining objects with a renderer, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage, and

based on a depth comparison involving the near z-values, conditionally reading from the second depth buffer z-values previously stored for image samples.

68. The method as recited in claim 67, wherein the stored near z-values are compared with far z-values computed for other objects, and the previously stored z-values are conditionally read based on the comparison.

69. The method as recited in claim 68, wherein the previously stored z-values are read from memory only if the far z-values computed for the other objects are farther than or equal to the corresponding near z-values.

70. The method as recited in claim 69, wherein results of the comparison are stored in a mask.

71. The method as recited in claim 70, wherein the mask is used in a decision to conditionally read from the memory the previously stored z-values.

72. The method as recited in claim 67, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against the first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and the renderer is positioned downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

73. A system for avoiding reading z-values in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) logic for a first pass including:

logic for transforming objects,

logic for creating an occlusion image in a first depth buffer requiring a first amount of storage, and

logic for storing near z-values in the occlusion image, each near z-value representative of a near z-value on one of the objects; and

(b) logic for a second pass including:

logic for conservatively culling objects utilizing the occlusion image, and

logic for rendering the remaining objects with a renderer, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage, and

logic for, based on a depth comparison involving the near z-values, conditionally reading from the second depth buffer z-values previously stored for image samples.

74. The system as recited in claim 73, wherein the stored near z-values are compared with far z-values computed for other objects, and the previously stored z-values are conditionally read based on the comparison.

75. The system as recited in claim 74, wherein the previously stored z-values are read from memory only if the far z-values computed for the other objects are farther than or equal to the corresponding near z-values.

76. The system as recited in claim 75, wherein results of the comparison are stored in a mask.

77. The system as recited in claim 76, wherein the mask is used in a decision to conditionally read from the memory the previously stored z-values.

78. The system as recited in claim 73, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against the first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and the renderer is positioned downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which

were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

79. A computer program product for avoiding reading z-values in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) computer code for a first pass including:

computer code for transforming objects,

computer code for creating an occlusion image in a first depth buffer requiring a first amount of storage, and

computer code for storing near z-values in the occlusion image, each near z-value representative of a near z-value on one of the objects; and

(b) computer code for a second pass including:

computer code for conservatively culling objects utilizing the occlusion image, and

computer code for rendering the remaining objects with a renderer, wherein the renderer requires a second depth buffer with a second amount of storage greater than the first amount of storage, and

computer code for, based on a depth comparison involving the near z-values, conditionally reading from the second depth buffer z-values previously stored for image samples.

80. A method for avoiding processing in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) during a first pass,

transforming an object,

determining whether at least a portion of the object has been culled, and

storing visibility information indicating whether the at least portion of the object has been culled; and

(b) during a second pass,

conditionally skipping processing in at least a portion of the graphics pipeline based on the visibility information.

81. The method as recited in claim 80, wherein the visibility information includes a visibility bit.

82. The method as recited in claim 80, wherein it is determined whether the at least portion of the object has been culled using z-value culling.

83. The method as recited in claim 80, wherein it is determined whether the at least portion of the object has been culled using stencil-value culling.

84. The method as recited in claim 80, wherein the processing of the at least portion of the object is skipped during the second pass if the visibility information indicates that the at least portion of the object has been culled.

85. The method as recited in claim 80, wherein it is determined whether the entire object has been culled, and the visibility information indicates whether the entire object has been culled.

86. The method as recited in claim 80, wherein during the first pass, the object is passed to a rasterizer for determining which of a plurality of tiles the object overlaps.

87. The method as recited in claim 86, wherein it is determined whether the object is culled in the at least one tile overlapped by the object, and the visibility information indicates whether the object has been culled.

88. The method as recited in claim 87, wherein the processing of the at least one tile is skipped if the visibility information indicates that the at least one tile has been culled.

89. The method as recited in claim 88, wherein the processing that is skipped includes reading of an occlusion image associated with the at least one tile.

90. A system for avoiding processing in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) logic for a first pass including:

logic for transforming an object,

logic for determining whether at least a portion of the object has been culled, and

logic for storing visibility information indicating whether the at least portion of the object has been culled; and

(b) logic for a second pass including:

logic for conditionally skipping processing in at least a portion of the graphics pipeline based on the visibility information.

91. The system as recited in claim 90, wherein the visibility information includes a visibility bit.

92. The system as recited in claim 90, wherein it is determined whether the at least portion of the object has been culled using z-value culling.

93. The system as recited in claim 90, wherein it is determined whether the at least portion of the object has been culled using stencil-value culling.

94. The system as recited in claim 90, wherein the processing of the at least portion of the object is skipped during the second pass if the visibility information indicates that the at least portion of the object has been culled.

95. The system as recited in claim 90, wherein it is determined whether the entire object has been culled, and the visibility information indicates whether the entire object has been culled.

96. The system as recited in claim 90, wherein during the first pass, the object is passed to a rasterizer for determining which of a plurality of tiles the object overlaps.

97. The system as recited in claim 96, wherein it is determined whether the object is culled in the at least one tile overlapped by the object, and the visibility information indicates whether the object has been culled.

98. The system as recited in claim 97, wherein the processing of the at least one tile is skipped if the visibility information indicates that the at least one tile has been culled.

99. The system as recited in claim 98, wherein the processing that is skipped includes reading of an occlusion image associated with the at least one tile.

100. A computer program product for avoiding processing in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) computer code for a first pass including:

computer code for transforming an object,

computer code for determining whether at least a portion of the object has been culled, and

computer code for storing visibility information indicating whether the at least portion of the object has been culled; and

(b) computer code for a second pass including:

computer code for conditionally skipping processing in at least a portion of the graphics pipeline based on the visibility information.

101. A method for avoiding processing in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) during a first pass,

transforming objects,

testing the objects for visibility,

determining whether a last object processed within an entity of a screen is entirely visible, and

storing status information indicating whether the last object processed within the entity of the screen is entirely visible; and

(b) during a second pass,

when rendering each object within the entity of the screen, avoiding writing and reading z-values to and from a z-buffer if the status information indicates that the last object processed within the entity of the screen was entirely visible.

102. The method as recited in claim 101, wherein the entity includes an image sample.

103. The method as recited in claim 101, wherein the entity includes a region of a tile.

104. The method as recited in claim 101, wherein the entity includes a tile.

105. The method as recited in claim 101, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against a first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and a renderer that is positioned downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

106. A system for avoiding processing in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) logic for a first pass including:

logic for transforming objects,

logic for testing the objects for visibility,

logic for determining whether a last object processed within an entity of a screen is entirely visible, and

logic for storing status information indicating whether the last object processed within the entity of the screen is entirely visible; and

(b) logic for a second pass including:

logic for, when rendering each object within the entity of the screen, avoiding writing and reading z-values to and from a z-buffer if the status information indicates that the last object processed within the entity of the screen was entirely visible.

107. The system as recited in claim 106, wherein the entity includes an image sample.

108. The system as recited in claim 106, wherein the entity includes a region of a tile.

109. The system as recited in claim 106, wherein the entity includes a tile.

110. The system as recited in claim 106, wherein the graphics pipeline includes a culling stage having an input for receiving a plurality of the objects, the culling stage testing the objects against a first depth buffer for occlusion and non-definitively but conservatively culling objects from the plurality of objects which it proves to be occluded; and a renderer that is positioned downstream of the culling stage which, while the culling stage conservatively culls objects for a given frame, renders objects into the given frame which were tested for occlusion in the culling stage but which were not proven upstream of the renderer to be occluded.

111. A computer program product for avoiding processing in a multi-pass rendering algorithm in a graphics pipeline, comprising:

(a) computer code for a first pass including:

computer code for transforming objects,

computer code for testing the objects for visibility,

computer code for determining whether a last object processed within an entity of a screen is entirely visible, and

computer code for storing status information indicating whether the last object processed within the entity of the screen is entirely visible; and

(b) computer code for a second pass including:

computer code for, when rendering each object within the entity of the screen, avoiding writing and reading z-values to and from a z-buffer if the status information indicates that the last object processed within the entity of the screen was entirely visible.